Theoretical Computer Science 114 (1993) 119–147
Elsevier

The expressive power of implicit specifications*

Kim Guldstrand Larsen

Department of Computer Science, Aalborg University, Aalborg, Denmark

Abstract

Larsen, K.G., The expressive power of implicit specifications, Theoretical Computer Science 114 (1993) 119–147.

In this paper we investigate the expressive power of *implicit specifications* of concurrent systems. That is, we consider specifications S specifying a system P not directly, but rather *in combination* with a given environment. Formally, we shall represent the environment as a CCS context of the form $(A|[\ ][\Phi])\backslash L$, where A is a process expression and $|$, $\backslash L$ and $[\Phi]$ are the parallel, restriction and renaming operators of CCS, respectively. We investigate the expressive power of the following three types of implicit specifications:

$$(A|P[\Phi])\backslash L \sim B, \tag{i}$$

where $\sim$ is strong bisimulation equivalence and B is a process expression;

$$(A|P[\Phi])\backslash L \lhd S, \tag{ii}$$

where S is a modal specification and $\lhd$ is refinement between modal specifications (a generalization of bisimulation equivalence); and

$$(A|P[\Phi])\backslash L \models F, \tag{iii}$$

where F is a formula of Hennessy–Milner logic extended with a (maximal) recursion construct. The paper offers the following two main results. When A, B and S are recursion-free and F is a Hennessy–Milner logic formula, the expressive powers of all three types of implicit specifications are the same and equal that of Hennessy–Milner logic. When A and S are regular, the expressive powers of implicit specifications of type (ii) and (iii) are the same and identical to that of Hennessy–Milner logic extended with (maximal) recursion. Finally, we recall from [21] that the expressive power of implicit specifications does not increase by modelling the environment as other contexts (of CCS or other process calculi).

Correspondence to: K.G. Larsen, Aalborg University, Department of Mathematics and Computer Science, Fredrik Bajersvej 7, 9220 Aalborg, Denmark. Email: kgl@iesd.auc.dk.

*This paper is a full version of a paper with the same title in the Proceedings of the 18th Colloquium on Automata, Languages and Programming, Madrid, 1991, Lecture Notes in Computer Science, Vol. 510. The work was supported in part by the Danish National Research Council under the DART project.

1. Introduction

In this paper we investigate the expressive power of implicit specifications of concurrent systems. That is, we consider specifications S which do not specify a system P directly, but rather the system in combination with a given environment. Process algebra provides an elegant way to represent such environments formally as contexts. Thus, an implicit specification of a system P is

$$C(P)\,\mathbf{sat}\,S, \tag{1}$$

where C is a context representing the given environment, and **sat** is a suitably chosen satisfaction relation.

Implicit specifications are inherent in any top-down design methodology. In such a methodology the designer will begin with a specification (S) of the system to be constructed, and proceed to design the structure (C) of an implementation. To achieve a final implementation, it remains to develop the subcomponents $P_1, \ldots, P_n$ of the chosen structure (perhaps using the design methodology recursively). Obviously, the requirements to these subcomponents $P_1, \ldots, P_n$ can now be formulated as an implicit specification:

$$C(P_1, \ldots, P_n)\,\mathbf{sat}\,S.$$

Here we restrict ourselves to the case $n=1$.

In process algebra the languages for specifications and implementations coincide and the sastisfaction relation is often given in terms of an equivalence relation (e.g. strong and weak bisimulation equivalence, testing equivalence, failure equivalence). Thus, *direct* specifications of systems tend to become overly explicit as the allowed implementations are limited to a single equivalence class. In order to allow for a variety of (possibly inequivalent) specifications, it is often convenient to resort to *implicit* specifications: e.g. in Milner's books on CCS [25, 28] the scheduler is given an implicit specification; also the observation criterion used in AUTO [8] corresponds very closely to the idea of implicit system specification. The obvious question to ask now is whether the use of implicit specifications is necessary or whether direct specifications are equally expressible (e.g. in the scheduler example of [25, 28] a direct specification is also provided). Generally speaking, we will show that implicit specifications do increase the expressive power, and we shall provide explicit characterizations of the power achieved.

More specifically, we shall consider implicit specifications based on CCS contexts of the form $(A|[\ ][\Phi])\backslash L$, where A is a process expression and $|$, $\backslash L$ and $[\Phi]$ are the parallel, restriction and renaming operators of CCS, respectively. This type of contexts is frequently encountered in a top-down development of systems in CCS. We investigate the expressive power of the following three types of implicit specifications:

$$(A|P[\Phi])\backslash L \sim B, \tag{2}$$

where $\sim$ is strong bisimulation equivalence and B is a process expression;

$$(A|P[\Phi])\backslash L \lhd S, \tag{3}$$

where S is a modal specification and $\lhd$ is refinement between modal specifications (a generalization of bisimulation equivalence); and

$$(A|P[\Phi])\backslash L \models F, \tag{4}$$

where F is a formula of Hennessy–Milner logic extended with a (maximal) recursion construct. The paper offers the following two main results. When A, B and S are finite and F is a Hennessy–Milner logic formula, the expressive powers of all three types of implicit specifications are the same and equal that of Hennessy–Milner logic. When A and S are regular, the expressive powers of implicit specifications of type (3) and (4) are the same and identical to that of Hennessy–Milner logic extended with (maximal) recursion. Finally, we recall from [21] that the expressive power of implicit specifications does not increase by modelling the environment as other contexts (of CCS or other process calculi).

Methods for synthesizing solutions to implicit specifications have been presented by Shields [33], Parrow [30], Lewis and Qin [23] and Merlin and Bochmann [24]. Larsen and Xinxin [22] consider general implicit specifications of the form $C(P) \sim B$ (thus generalizing (2)), where C can be an arbitrary[1] context. A characterization of the solutions is given, based on which a single-exponential time decision algorithm (i.e. an algorithm for deciding whether there exist solutions) is induced.

Common to all proposed methods [33, 30, 22–24] is that the proposed algorithms require exponential time, even in spite of the fact that restrictions on the involved contexts and processes are often imposed. In [15], lower bounds for the synthesis problem[2] have been established. In particular, it has been shown that the synthesis of solutions to implicit specifications of the forms (2) and (3) are PSPACE-hard in the size of A and B. This paper's characterization of the expressive power of certain classes of implicit specifications originates from attempts at improving this lower bound. In fact, based on the results we obtain, we can improve one of the lower bounds: the synthesis of solutions to implicit specification of the form (3) is DEXPTIME-complete.

In the next section, we introduce the notion of specification formalism and formalize the concepts of implicit specification and expressiveness. In Section 3 we present the basic specification formalisms: modal transition systems and (recursive) Hennessy–Milner logic. In Section 4 we introduce a new parallel operator and relate the implicit specifications it generates to those based on CCS and LOTOS parallel operators. In Section 5 we demonstrate that the expressive power of implicit specifications induced by finite specification formalisms is characterized by Hennessy–Milner logic. In Section 6 we demonstrate that the expressive power of

[1] C should be describable as an *action transducer*; see [22, 21].

[2] Or rather the associated decision problem concerned with the *existence* of solutions.

implicit specifications induced by regular modal specifications is characterized by recursive Hennessy–Milner logic. Finally, we discuss open problems and future work. The appendices contain the detailed proofs.

2. Implicit specifications and expressiveness

To provide a basic terminology for the remainder of this paper, we introduce in this section the notion of specification formalism and formalize the concepts of (induced) implicit specification formalism and (relative) expressiveness.

Assume that Π is a set of processes or implementations. Then a *specification formalism* for Π is a structure $\mathscr{S} = \langle \Sigma, \mathbf{sat} \rangle$, where Σ is some set of specifications and **sat** is a subset of $\Pi \times \Sigma$. Whenever P **sat** S for P a process and S a specification, we say that P satisfies S, or P is correct with respect to S; the relation **sat** is referred to as the satisfaction relation.

A specification formalism $\mathscr{S} = \langle \Sigma, \mathbf{sat} \rangle$ induces the two functions $\mathbf{Mod}_{\mathscr{S}} : \Sigma \to 2^{\Pi}$ and $\mathbf{Th}_{\mathscr{S}} : \Pi \to 2^{\Sigma}$ relating specifications and processes:

$$\mathbf{Mod}_{\mathscr{S}}(S) =^{\Delta} \{P \in \Pi \mid P \,\mathbf{sat}\, S\}, \qquad \mathbf{Th}_{\mathscr{S}}(P) =^{\Delta} \{S \in \Sigma \mid P \,\mathbf{sat}\, S\},$$

where $S \in \Sigma$ and $P \in \Pi$. Thus, $\mathbf{Mod}_{\mathscr{S}}(S)$ is the set of all processes satisfying S, and $\mathbf{Th}_{\mathscr{S}}(P)$ is the set of all specifications which are satisfied by P. Both $\mathbf{Mod}_{\mathscr{S}}$ and $\mathbf{Th}_{\mathscr{S}}$ can be extended to sets by intersecting the contributions of the elements, e.g., for a set of specifications A, $\mathbf{Mod}_{\mathscr{S}}(A) = \bigcap_{S \in A} \mathbf{Mod}_{\mathscr{S}}(S)$. A specification S is *consistent* or *sastisfiable* if the set of models $\mathbf{Mod}_{\mathscr{S}}(S)$ is nonempty.

Refinement between specifications is now defined as just logical implication or, equivalently, as the inclusion between their models:

$$S_1 \Rightarrow S_2 \iff^{\Delta} \mathbf{Mod}_{\mathscr{S}}(S_1) \subseteq \mathbf{Mod}_{\mathscr{S}}(S_2),$$

i.e. S_1 refines S_2 provided any process satisfying S_1 also satisfies S_2. Dually, an equivalence is induced on processes based on their theories: two processes are equivalent in case they satisfy the same specifications:

$$P \equiv_{\mathscr{S}} Q \iff^{\Delta} \mathbf{Th}_{\mathscr{S}}(P) = \mathbf{Th}_{\mathscr{S}}(Q).$$

The *relative expressivity* between specification formalisms can be formalized in the following way: For two specification formalisms $\mathscr{S} = \langle \Sigma, \mathbf{sat} \rangle$ and $\mathscr{S}' = \langle \Sigma', \mathbf{sat}' \rangle$ over a process set Π, we say that $\mathscr{S}$ is at least as expressive as $\mathscr{S}'$ if, for any specification of $\mathscr{S}'$, one can find a specification of $\mathscr{S}$ allowing the same implementations, i.e.

$$\forall S' \in \Sigma'.\ \exists S \in \Sigma.\ \mathbf{Mod}_{\mathscr{S}}(S) = \mathbf{Mod}_{\mathscr{S}'}(S').$$

We shall use the notation $\mathscr{S}' \to \mathscr{S}$ to indicate that $\mathscr{S}$ is at least as expressive as $\mathscr{S}'$. Note that $\to$ constitutes a preorder between specification formalisms.

Given a set of (unary) contexts $\mathscr{C}$ – i.e. whenever $C \in \mathscr{C}$, $C : \Pi \to \Pi$ – and a specification formalism $\mathscr{S} = \langle \Sigma, \mathbf{sat} \rangle$, a new specification formalism $\mathscr{I}[\mathscr{C}, \mathscr{S}]$ of *implicit*

specifications is induced. More precisely,

$$\mathscr{I}[\mathscr{C}, \mathscr{S}] = \langle \mathscr{C} \times \Sigma, \mathbf{sat}_{\mathscr{I}} \rangle,$$

where

$$P \, \mathbf{sat}_{\mathscr{I}} (C, S) \Leftrightarrow^{\Delta} C(P) \, \mathbf{sat} \, S,$$

i.e. P satisfies (C, S) in case the combined process $C(P)$ satisfies S.

The purpose of the remainder of this paper is to study the relative expressiveness of various specification formalisms and the implicit formalisms they induce, given certain context families.

3. Processes and specification formalisms

3.1. Processes

In this paper we follow the *reactive* view of concurrent systems advocated by Pnueli [32], i.e. we assume that the semantics of concurrent systems (or *processes*) is given in terms of their interaction with their environment using the well-established model of *labelled transition systems* [31].

Definition 3.1. A *labelled transition system* is a structure $\mathscr{P} = (\Pi, A, \rightarrow)$, where Π is a set of processes (states or configurations), A is a set of actions and $\rightarrow \subseteq \Pi \times A \times \Pi$ is the transition relation.

For $(P, a, Q) \in \rightarrow$, we shall usually write $P \xrightarrow{a} Q$, which is to be interpreted: "P may perform the action a and become Q in doing so." We refer to Q as a (a-) derivative of P. We shall write $P \xrightarrow{a}$ as an abbreviation for $\exists Q.\ P \xrightarrow{a} Q$.

The notion of *bisimulation* [29, 26] provides a means of identifying processes based on their operational behaviour.

Definition 3.2. Let $\mathscr{P} = (\Pi, A, \rightarrow)$ be a labelled transition system. A *bisimulation* B is a binary relation on Π such that, whenever $(P, Q) \in B$ and $a \in A$, the following holds:

(1) Whenever $P \xrightarrow{a} P'$, $Q \xrightarrow{a} Q'$ for some Q', with $(P', Q') \in B$.

(2) Whenever $Q \xrightarrow{a} Q'$, $P \xrightarrow{a} P'$ for some P', with $(P', Q') \in B$.

P is said to be bisimilar to Q in case (P, Q) is contained in some bisimulation B. We write $P \sim Q$ in this case.

3.2. Behavioural specifications

Modal transition systems are a generalization of classical labelled transition systems. In particular, modal transition systems provide a (graphical) specification

formalism for processes (expressed through labelled transition systems) and are studied at length in [20, 14, 19, 4, 22]. The specifications expressible using modal transition systems (modal specifications) typically impose restrictions on the transitions of possible implementations by telling which transitions are *necessary* and which are *admissible*. This is reflected by the structure of a modal transition system which contains *two* transition relations: $\rightarrow_{\square}$ for describing the *required* transitions and $\rightarrow_{\diamond}$ for describing the *allowed* transitions.

Definition 3.3. A *modal transition system* is a structure $\mathscr{S}=(Q, A, \rightarrow_{\square}, \rightarrow_{\diamond})$, where Q is a set of (modal) specifications, A is a set of actions and $\rightarrow_{\square}, \rightarrow_{\diamond} \subseteq Q \times A \times Q$ are two transition relations satisfying the condition $\rightarrow_{\square} \subseteq \rightarrow_{\diamond}$.

The condition $\rightarrow_{\square} \subseteq \rightarrow_{\diamond}$ says that anything required is also allowed, ensuring that any modal specification is consistent. Recall that the behaviour of processes is given in terms of a standard labelled transition system $P=(\Pi, A, \rightarrow)$. Hence, we may view processes as modal specifications, where all requirements are necessary ones, by considering processes as elements of the derived modal transition system $\mathscr{S}_{\mathscr{P}}=(\Pi, A, \rightarrow_{\square}, \rightarrow_{\diamond})$, with $\rightarrow_{\square}=\rightarrow_{\diamond}=\rightarrow$.

Now, the more a modal specification requires and the less it allows, the stronger we expect the specification to be. Using the derivation relations $\rightarrow_{\square}$ and $\rightarrow_{\diamond}$, this may be formalized by the following notion of *refinement*.

Definition 3.4. Let $\mathscr{S}_{\mathscr{P}}=(\Pi, A, \rightarrow_{\square}, \rightarrow_{\diamond})$ be a modal transition system. A *refinement* R is a binary relation on Q such that, whenever $(S, T) \in R$ and $a \in A$, the following holds:

(1) Whenever $S \xrightarrow{a}_{\diamond} S'$, $T \xrightarrow{a}_{\diamond} T'$ for some T', with $(S', T') \in R$.

(2) Whenever $T \xrightarrow{a}_{\square} T'$, $S \xrightarrow{a}_{\square} S'$ for some S', with $(S', T') \in R$.

S is said to be a refinement of T in case (S, T) is contained in some refinement R. We write $S \lhd T$ in this case.

A straightforward generalization allows us to compare specifications from different modal transition systems (essentially by applying the above definition to disjoint sums of modal transition systems). In particular, if $P \lhd S$, where P is a process (viewed as a specification through the derived modal transition system) and S is a specification, we say that P is an *implementation* of S.

Now, the defined refinement relation $\lhd$ enjoys many pleasant properties: $\lhd$ is itself a refinement; in fact, the maximal one. Also, $\lhd$ is a *preorder* (reflexive and transitive) allowing looseness in specifications. As an example, the weakest modal specification $\mathscr{U}$ is one which constantly allows any action but never requires that any action must be performed. Operationally, $\mathscr{U}$ is completely defined by $\mathscr{U} \xrightarrow{a}_{\diamond} \mathscr{U}$ for all actions a. Moreover, the notion of refinement is a generalization of that of bisimulation [29, 26, 28]. If $\rightarrow_{\square}=\rightarrow_{\diamond}$ (e.g. when the modal transition system is derived from a process

system) the notions of refinement and bisimulation coincide, and $\lhd$ becomes bisimulation equivalence $\sim$.

Example 3.5. An a-sender is a process that will never refuse to perform the action a as long as the observer asks for nothing else. Examples of a-senders are the processes P_1 and P_2 in Fig. 1, whereas P_3 and P_4 are examples of processes not being a-senders (they may both refuse to do a after one a action). The notion of an a-sender may be specified operationally as S in Fig. 2. The transition $S \xrightarrow{a}_{\Box} S$ guarantees that any implementation can perform a and become an a-sender. The transition $S \xrightarrow{b}_{\Diamond} \mathscr{U}$, $b \neq a$, allows further an implementation to perform any action different from a, after which no restriction is imposed (indicated by $\mathscr{U}$). Now, it is perfectly simple to construct refinements proving $P_1 \lhd S$ and $P_2 \lhd S$. Also, it may easily be argued that $\neg(P_3, P_4 \lhd S)$. A similar but slightly wider class of processes is that of a-transmitters, consisting of all processes possessing an infinite a-computation. Thus, besides P_1 and P_2, also P_3 is an a-transmitter, whereas P_4 is not. Operationally, we may specify the concept of an a-transmitter as T in Fig. 2. Clearly, T is more liberal than S since it has the additional a-transition $T \xrightarrow{a}_{\Diamond} \mathscr{U}$, allowing an implementation to perform a without necessarily becoming an a-transmitter again. It is easily proved that $\{(S, T), (\mathscr{U}, \mathscr{U})\}$ is a refinement and, hence, that $S \lhd T$.

We now present the language for *regular modal specifications* from [20], extending the language of regular processes (as in [27]), by a simple addition of modalities to the prefixing construct. We assume a set of specification variables ranged over by $x, y, z, \ldots$ and let $S, S_0, S_1, \ldots, T, T_0, \ldots$, range over specifications. Also, let a range over actions. Then the set of regular modal specifications is generated by the following grammar:

$$S ::= \text{Nil} \mid \mathscr{U} \mid a_{\Diamond}.S \mid a_{\Box}.S \mid S_0 + S_1 \mid x \mid \text{rec}x.S.$$

The concepts of free and bound variables are defined as usual; in particular, we call a regular modal specification *closed* if it contains no free variables. We shall use the

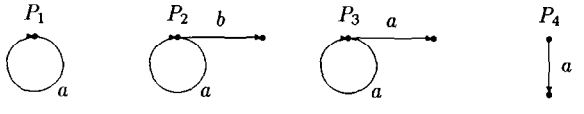

Fig. 1. Potential implementations.

Fig. 2. Specifications of an a-sender and an a-transmitter.

standard notation $T[S_1/x_1, \ldots, S_n/x_n]$ to describe the specification obtained by simultaneously replacing in T all free occurrences of a variable x_i with the corresponding specification S_i (with bound variables of T being renamed when capturing of free variables can occur).

We denote by $\mathscr{R}$ the set of all expressions generated by the above grammar, and by $\mathscr{R}_c$ the set of all closed expressions (called regular modal specifications); by $\mathscr{T}$ we denote the set of all *total* specifications of $\mathscr{R}_c$, i.e. specifications which use neither $\mathscr{U}$ nor $a_\diamond$.-prefixing (and, hence, can be considered as *regular processes*). For expressions in $\mathscr{T}$ we shall often use the more standard process notation $a.P$ instead of $a_\Box.P$. Specifications of $\mathscr{R}_c$ which do not use the rec-construct are called *finite* and are denoted by $\mathscr{F}$. $\mathscr{T}F$ denotes the set of modal specifications which are both total and finite (and, hence, corresponds to *finite processes*). The following table summarizes the various specification classes.

Abbreviation	Specification class
$\mathscr{R}$	*All* specification expressions
$\mathscr{R}_c$	*Closed* expressions (regular modal specifications)
$\mathscr{T}$	*Total* specifications (no use of $\mathscr{U}$ or $a_\diamond$.)
$\mathscr{F}$	*Finite* specifications (no use of rec)
$\mathscr{T}F$	*Finite* and *total* specifications

The operational semantics of regular modal specifications is now given in terms of a modal transition system.

Definition 3.6. Let $\rightarrow_\diamond, \rightarrow_\Box$ be the smallest subsets of $\mathscr{R}_c \times A \times \mathscr{R}_c$ closed under the following rules:

$$\mathscr{U}: \mathscr{U} \xrightarrow{a}_\diamond \mathscr{U},$$

$$\diamond: a_\diamond.S \xrightarrow{a}_\diamond S,$$

$$\Box: a_\Box.S \xrightarrow{a}_\diamond S,$$

$$a_\Box.S \xrightarrow{a}_\Box S,$$

$$+: \frac{S_1 \xrightarrow{a}_m S_1'}{S_1 + S_2 \xrightarrow{a}_m S_1'}, \quad \frac{S_2 \xrightarrow{a}_m S_2'}{S_1 + S_2 \xrightarrow{a}_m S_2'},$$

$$\text{rec}: \frac{S[\text{rec}x.S/x] \xrightarrow{a}_m T}{\text{rec}x.S \xrightarrow{a}_m T},$$

where m ranges over $\Box$ and $\diamond$.

It is easily proved that $\rightarrow_{\sqsubset} \subseteq \rightarrow_{\diamond}$, ensuring that $(\mathscr{R}, A, \rightarrow_{\sqsubset}, \rightarrow_{\diamond})$ is indeed a modal transition system.

Example 3.7. Assuming that the action set A is finite, we may express the specifications $\mathscr{U}$ and S and T from Example 3.5 as follows:

$$\mathscr{U} = \mathrm{rec}x.\left(\sum_a a_{\diamond}.x\right),$$

$$S = \mathrm{rec}x.\left(a_{\sqsubset}.x + \sum_{b \neq a} b_{\diamond}.\mathscr{U}\right),$$

$$T = \mathrm{rec}x.(a_{\sqsubset}.x + \mathscr{U}).$$

$\mathscr{R}_c$ (and, similarly, $\mathscr{T}$, $\mathscr{F}$ and $\mathscr{T}F$), together with the refinement relation, $\lhd$ constitutes a specification formalism $\langle \mathscr{R}_c, \lhd \rangle$ which we (ambiguously) shall refer to as $\mathscr{R}_c$. Note that S_1 refines S_2 in case $S_1 \lhd S_2$.

3.3. Logical specifications

Hennessy and Milner presented in [11] a simple propositional modal logic (which we shall refer to as Hennessy–Milner logic) describing properties of processes. As a main theorem, they showed that two processes are bisimilar just in case they satisfy the same properties of this logic (subject to a technical condition called image-finiteness).

The modal μ-calculus of Kozen [16] can be seen as an extension of Hennessy–Milner logic with the ability to define properties recursively. In [9, 17] it is indicated that the resulting logic provides a very expressive specification language: in fact, all the standard operators from branching time temporal logic [2] are derivable, allowing *safety* as well as *liveness* properties of processes to be expressed conveniently. A number of papers have recently been written on the subject of model checking with respect to the modal μ-calculus or related logics [9, 17, 34, 35, 1, 5–7].

Here we consider a negation-free version of Hennessy–Milner logic extended with a ν operator, allowing properties to be defined recursively (as *maximal* fixed points). As such, we do not obtain the full expressive power of the modal μ-calculus but are only able to express safety properties within the logic. However, the logic introduced is interesting since it turns out to characterize precisely the expressive power of implicit specifications based on regular modal specifications.

We assume a set V of formula variables ranged over by $x, y, z, \ldots$ and let $F, F_1, F_2, \ldots, G, G_1, G_2, \ldots$ range over formula expressions. Then the set of formulas $\mathscr{L}$ is generated by the following grammar:

$$F ::= \mathrm{tt} \mid \mathrm{ff} \mid F \wedge G \mid F \vee G \mid \langle a \rangle F \mid [a]F \mid x \mid \nu x.F.$$

The concepts of free and bound variables are defined as usual, with ν providing the binding construct. In particular, we call a formula *closed* if it contains no free variables and denote by $\mathscr{H}_\nu$ the set of closed formulas. The sest of closed formulas which do not use the recursion construct ν coincides with the classical Hennessy–Milner logic and is denoted by $\mathscr{H}$. We shall use the standard notation $F[G_1/x_1, \ldots, G_n/x_n]$ to describe the simultaneous substitution of G_i for the variable x_i (with bound variables of F being renamed when capturing of variables of G can occur).

The semantics of a closed formula F is now given as the set of processes $[\![F]\!]$ *satisfying* the formula. However, as formulas, in general, may contain free variables, we define their semantics relative to an environment $\sigma : V \to 2^{\Pi}$ giving the semantics of free variables. As is standard, we use the notation $\sigma\{U/x\}$, where $U \subseteq \Pi$, to denote the environment which is identical to σ except that the set U is returned for the variable x. For F a formula of $\mathscr{L}$ and σ an environment, we now define the set of processes $[\![F]\!]\sigma$ inductively as follows:

(i) $[\![x]\!]\sigma = \sigma(x)$,
(ii) $[\![\mathrm{tt}]\!]\sigma = \Pi$,
(iii) $[\![\mathrm{ff}]\!]\sigma = \emptyset$,
(iv) $[\![F \wedge G]\!]\sigma = [\![F]\!]\sigma \cap [\![G]\!]\sigma$,
(v) $[\![F \vee G]\!]\sigma = [\![F]\!]\sigma \cup [\![G]\!]\sigma$,
(vi) $[\![\langle a \rangle F]\!]\sigma = \{P \mid \exists P' \in [\![F]\!]\sigma.\ P \xrightarrow{a} P'\}$,
(vii) $[\![[a]F]\!]\sigma = \{P \mid \forall P'.\ P \xrightarrow{a} P' \Rightarrow P' \in [\![F]\!]\sigma\}$,
(viii) $[\![\nu x.F]\!]\sigma = \bigcup\{U \mid U \subseteq [\![F]\!]\sigma\{U/x\}\}$.

$[\![F]\!]\sigma$ only depends on the part of σ which concerns the free variables of F. In particular, if F is closed, σ is immaterial, and we will simply write $[\![F]\!]$ for $[\![F]\!]\sigma$, where σ could be any environment. For F being a closed formula and P a process, we shall use the notation $P \models F$ for $P \in [\![F]\!]$. Now the set of formulas $\mathscr{H}_\nu$ (and, similarly, $\mathscr{H}$) determines, together with $\models$, a specification formalism $\langle \mathscr{H}_\nu, \models \rangle$, which we (ambiguously) shall denote by $\mathscr{H}_\nu$.

The following few examples demonstrate the type of safety properties expressible in $\mathscr{H}_\nu$. We refer the interested reader to [17] for more information.

Example 3.8. The following formula expresses that any sequence of a-transitions from a process P (satisfying the formula) will lead to a state satisfying the formula F:

$$\nu x.F \wedge [a]x.$$

The next formula is satisfied by a process P just in case any finite a-computation (i.e. maximal sequence of a-transitions) has even length:

$$\nu x.[a](\langle a \rangle \mathrm{tt} \wedge [a]x).$$

Finally, the existence of an infinite a-computation (i.e. the concept of an a-transmitter in Example 3.5) can be expressed as

$$\nu x.\langle a \rangle x.$$

3.4. Results of expressiveness

In Fig. 3 we have graphically visualized the complete order of relative expressiveness between the basic specification formalisms introduced in this section. The relationships between the behavioural specification formalisms (and, similarly, between the two logical formalisms) are completely determined by language containment. The relationship $\mathscr{F} \rightarrow \mathscr{H}$ was proved in [4], where it was also demonstrated that $\mathscr{H}$ is strictly more expressive than $\mathscr{F}$: in terms of expressive power, $\mathscr{F}$ was shown to be identical to a sublogic of $\mathscr{H}$ (consisting of all *prime* formula).

Example 3.9. The modal specification $a_{\diamond}.(b_{\sqcup}.\mathscr{U}+\mathscr{U})+a_{\diamond}.(c_{\sqcap}.\mathscr{U}+\mathscr{U})$ corresponds to the Hennessy–Milner logic formula $[a](\langle b\rangle \mathrm{tt} \vee \langle c\rangle \mathrm{tt}) \wedge [b]\mathrm{ff} \wedge [c]\mathrm{ff}$. No modal specification corresponds to the (nonprime) formula $\langle a\rangle \mathrm{tt} \vee \langle b\rangle \mathrm{tt}$.

Similarly, $\mathscr{R}_c \rightarrow \mathscr{H}_v$ was proved in [19]. Here we illustrate the result through an example.

Example 3.10. The class of a-transmitters $\mathrm{rec}x.a_{\sqcap}.x+\mathscr{U}$ corresponds to the following recursive Hennessy–Milner logic formula: $\nu x.\langle a\rangle x$.

4. Contexts and context families

The main results of this paper, as claimed in the introduction, concern the expressive power of implicit specifications using CCS contexts of the form $(A \mid [\ ][\Phi])\backslash L$. In fact, we will obtain these results as (easy) corollaries of similar expressiveness results for implicit specifications involving contexts based on a new dyadic (parallel) operator $\gg$. In this section we will introduce this new operator $\gg$, and relate it to the well-known (parallel) operators of CCS [25, 28], CSP [12] and LOTOS [3].

Similar to the parallel operator of CCS, the dyadic operator $\gg$ requires a certain structure of the action set A. More specifically, for any action $a \in A$, there must exist distinct actions $a^+ \in A$ and $a^\circ \in A$ (+ and $\circ$ are postfix tagging symbols). If A_b is a set of basic actions, then the action set A, defined as the least set sastisfying

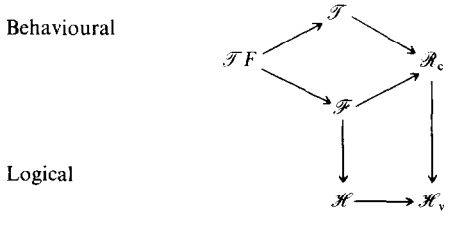

Fig. 3. Basic results of relative expressiveness.

$A = A_b \cup \{a^+ \mid a \in A\} \cup \{a^\circ \mid a \in A\}$, is an extension of A_b with the required property. We shall denote by A^+ the set of actions $\{a^+ \mid a \in A\}$ and, similarly, by A° the set of actions $\{a^\circ \mid a \in A\}$.

The dyadic process operator $\gg$ is now introduced with the following operational semantics:

$$\frac{P \xrightarrow{a} P'}{P \gg Q \xrightarrow{a} P' \gg Q}, \qquad \frac{P \xrightarrow{a^+} P' \quad Q \xrightarrow{a} Q'}{P \gg Q \xrightarrow{a} P' \gg Q'}.$$

Thus, in any transition of $P \gg Q$ the left component P must always participate. Whether or not the right component Q is to participate is determined by the tagging of P's actions:[3] $+$ indicates that Q has to participate, and $\circ$ indicates that Q cannot participate.

The operator $\gg$ satisfies a number of pleasant algebraic laws with respect to bisimulation equivalence, the (easy) proofs of which we leave to the reader:

$$(a^\circ.P \gg Q) \sim a.(P \gg Q), \tag{5}$$

$$(a^+.P \gg a.Q) \sim a.(P \gg Q), \tag{6}$$

$$(P + R) \gg Q \sim (P \gg Q) + (R \gg Q), \tag{7}$$

$$a^+.P \gg (R + Q) \sim (a^+.P \gg R) + (a^+.P \gg Q), \tag{8}$$

$$\text{Nil} \gg Q \sim \text{Nil}. \tag{9}$$

To compare the new operator $\gg$ with more well-known parallel operators, let us first recall (one of) the parallel operators of TCSP [13] and LOTOS [3], $\|_B$.[4] This operator – parameterized with an action set $B \subseteq A$ on which the two processes are required to synchronize – is described semantically by the following three inference rules:

$$\frac{P \xrightarrow{a} P'}{P \|_B Q \xrightarrow{a} P' \|_B Q} \ (a \notin B), \qquad \frac{Q \xrightarrow{a} Q'}{P \|_B Q \xrightarrow{a} P \|_B Q'} \ (a \notin B),$$

$$\frac{P \xrightarrow{a} P' \quad Q \xrightarrow{a} Q'}{P \|_B Q \xrightarrow{a} P' \|_B Q'} \ (a \in B).$$

For $B = A$, the LOTOS parallel operator becomes the parallel operator $\|$ of CSP [12]. Note that for this operator only the third rule is applicable.

[3] Following the operational semantic framework of [21, 22], we may alternatively describe unary contexts of the form $P \gg [\ ]$ as *action transducers*.

[4] In LOTOS the operator is written as $|[B]|$.

Finally, we assume a unary renaming operator $[\Phi]$ on processes for any renaming function $\Phi: A \to A$ with the following standard semantics:

$$\frac{P \xrightarrow{a} P'}{P[\Phi] \xrightarrow{\Phi a} P'[\Phi]}.$$

We shall denote by A^+/A the renaming function $\lambda a \in A.a^+$ and, similarly, by A°/A the renaming function $\lambda a \in A.a^\circ$. By $A/A^+ \cup A^\circ$ we denote the untagging renaming function, which maps tagged actions a^+ and a° to a, and acts as an identity function on untagged actions.

The following three algebraic laws (with respect to $\sim$) show how $P \gg Q$ degenerates to well-known constructs for special cases of P:

$$P[A^+/A] \gg Q \sim P \| Q, \tag{10}$$

$$P[A^\circ/A] \gg Q \sim P, \tag{11}$$

$$\left(\text{rec}x. \sum_{a \in B} a^+.x\right) \gg Q \sim Q, \tag{12}$$

where, in (12), sort(Q) must be contained in B. Here sort(Q) is a superset of the actions which may be performed in any computation of Q. sort(Q) is normally defined on the structure of Q in obvious ways. Using the LOTOS parallel operator together with renaming, we can derive the new operator $\gg$ by the following law:

$$P \gg Q \sim (P \|_{A^+ \setminus A^\circ} Q[A^+/A])[A/A^+ \cup A^\circ]. \tag{13}$$

Again we leave the quite straightforward proofs to the reader.

Turning to CCS, it does not seem possible to express the new parallel operator $\gg$ as a derived operator in terms of (CCS) parallel composition and renaming. However, the following lemma provides the basis for comparing the expressive power of implicit specifications based on the two types of parallel operators.

Lemma 4.1. *For any processes P and B, there exist processes $P_\uparrow$ and $B_\downarrow$ such that, for any process Q,*

$$P \gg Q \sim B \Leftrightarrow (P_\uparrow | Q[A^+/A]) \setminus A^+ \sim B_\downarrow.$$

Proof. The behaviour of $P_\uparrow$ and $B_\downarrow$ is given by the following inference rules:

$$\frac{P \xrightarrow{a^+} P'}{P_\uparrow \xrightarrow{\overline{a^+}} a^\circ.P'_\uparrow}, \qquad \frac{P \xrightarrow{a^-} P'}{P_\uparrow \xrightarrow{\tau} a^\circ.P'_\uparrow}, \qquad \frac{B \xrightarrow{a} B'}{B_\downarrow \xrightarrow{\tau} a^\circ.B'_\downarrow}.$$

With these definitions it can be shown that any transition of $P \gg Q$ corresponds to a sequence of two transitions of $(P_\uparrow | Q[A^+/A]) \backslash A^+$ in the following way:

$$P \gg Q \xrightarrow{a} P' \gg Q'$$

$$\Leftrightarrow (P_\uparrow | Q[A^+/A]) \backslash A^+ \xrightarrow{\tau} (a^\circ . P'_\uparrow | Q'[A^+/A]) \backslash A^+$$

$$\xrightarrow{a} (P'_\uparrow | Q'[A^+/A]) \backslash A^+.$$

In establishing this correspondence, two cases must be considered, depending upon the inference rule used for the transition of $P \gg Q$. As the transitions of B and $B_\downarrow$ are obviously related in the very same manner, the lemma follows. □

Based on the three parallel operators $\gg$, $|$ and $\|_B$, we introduce in Table 1 six families of unary contexts. We can now restate the algebraic law (13) and Lemma 4.1 as results of relative expressive power between the various implicit specification formalisms.

Theorem 4.2. *For $\mathscr{X} = \mathscr{T}$ and $\mathscr{X} = \mathscr{T}F$, the following relations hold:*

$$\mathscr{I}[\text{NEW}_\mathscr{X}, \mathscr{X}] \to \mathscr{I}[\text{CCS}_\mathscr{X}, \mathscr{X}], \qquad \mathscr{I}[\text{NEW}_\mathscr{X}, \mathscr{X}] \to \mathscr{I}[\text{LOTOS}_\mathscr{X}, \mathscr{X}].$$

Proof. In Lemma 4.1 simply note that $P_\uparrow, B_\downarrow \in \mathscr{T}F$ ($\in \mathscr{T}$) whenever $P, B \in \mathscr{T}F$ ($\in \mathscr{T}$). □

5. Implicit finite specifications

In this section we will demonstrate that implicit specifications induced by finite specification formalisms (i.e. $\mathscr{T}F$, $\mathscr{F}$ and $\mathscr{H}$) and families of finite contexts (i.e. $\text{PAR}_{\mathscr{T}F}$,

Table 1
Families of contexts

Context family	Contexts form
$\text{NEW}_\mathscr{T}$	$(Q \gg [\ ])$, where $Q \in \mathscr{T}$
$\text{NEW}_{\mathscr{T}F}$	$(Q \gg [\ ])$, where $Q \in \mathscr{T}F$
$\text{CCS}_\mathscr{T}$	$(Q \mid [\ ][\Phi]) \backslash L$, where $Q \in \mathscr{T}$
$\text{CCS}_{\mathscr{T}F}$	$(Q \mid [\ ][\Phi]) \backslash L$, where $Q \in \mathscr{T}F$
$\text{LOTOS}_\mathscr{T}$	$(Q \|_B [\ ][\Phi])[\Psi]$, where $Q \in \mathscr{T}$
$\text{LOTOS}_{\mathscr{T}F}$	$(Q \|_B [\ ][\Phi])[\Psi]$, where $Q \in \mathscr{T}F$

with PAR ranging over NEW, CCS and LOTOS; see Table 1) all coincide with respect to their expressive power. In fact, their expressive power is precisely that of Hennessy–Milner logic $\mathscr{H}$. We have visualized these results of relative expressiveness in Fig. 4. From [18, 21] we know that $\mathscr{I}[\mathscr{C},\mathscr{H}]\to\mathscr{H}$ holds for any family of unary contexts describable as finite-state *action transducers*. This condition is easily seen to be satisfied by all the families $\text{NEW}_{\mathscr{T}F}$, $\text{CCS}_{\mathscr{T}F}$ and $\text{LOTOS}_{\mathscr{T}F}$. Due to the expressiveness results in Theorem 4.2, it then suffices to consider the case PAR = NEW.

The relative strength of expressiveness between $\mathscr{T}F, \mathscr{F}$ and $\mathscr{H}$ induces a similar relationship between the corresponding implicit specification formalisms, as indicated in the figure.[5]

Also, each basic specification formalism is no more expressive than the corresponding implicit specification formalism. As an example, let $R\in\mathscr{T}F$, with depth n.[6] Then, for any process P, the following equivalence holds:

$$P\sim R \;\Leftrightarrow\; (U^{n+1}\gg P)\sim R,$$

where $U^0=\text{Nil}$ and $U^{n+1}=\sum_a a^+.U^n$. Note the close resemblance to law (12).

In [18] we proved that $\mathscr{H}\to\mathscr{I}[\mathscr{C},\mathscr{F}]$, where $\mathscr{C}$ was a "sufficiently rich" family of unary contexts. Figure 4 indicates that $\mathscr{C}=\text{PAR}_{\mathscr{T}F}$, with PAR ranging over NEW, CCS and LOTOS, will be such a "sufficiently rich" family. Here we show the even stronger (and considerably more difficult) result:

$$\mathscr{H}\to\mathscr{I}[\text{NEW}_{\mathscr{T}F},\mathscr{T}F],$$

from which the existing result of [18] follows as a corollary due to the transitivity of $\to$. That is, we establish that, for any Hennessy–Milner logic formula F, there exist total and finite processes Q_F and P_F such that, for all processes P, the following holds:

$$P\models F \;\Leftrightarrow\; (Q_F\gg P)\sim P_F.$$

First we observe in the following lemma that, for $Q\in\mathscr{T}F$, processes of the form $Q\gg P$ may be divided into a finite collection of equivalence classes with respect to $\sim$.

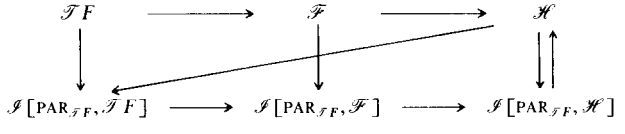

Fig. 4. Relative expressiveness I.

[5] For any context family $\mathscr{C}$, it is easy to see that $\mathscr{I}[\mathscr{C},\mathscr{S}]$ is monotonic in $\mathscr{S}$ with respect to $\to$.

[6] The depth of a specification $R\in\mathscr{T}F$ is defined structurally as $\text{depth}(\text{Nil})=0$, $\text{depth}(a_{\sqcup}.S=1+\text{depth}(S)$ and $\text{depth}(S_0+S_1)=\max\{\text{depth}(S_0),\text{depth}(S_1)\}$.

Lemma 5.1. *For any $Q \in \mathcal{T}F$, there is a finite set* $\mathrm{Beh}(Q) \subseteq \mathcal{T}F$ *such that*

$$\forall P.\exists R \in \mathrm{Beh}(Q).R \sim (Q \gg P),$$

$$\forall R \in \mathrm{Beh}(Q).\exists P.R \sim (Q \gg P),$$

$$\forall R_1, R_2 \in \mathrm{Beh}(Q).R_1 \neq R_2 \Rightarrow R_1 \not\sim R_2.$$

Proof. A simple proof by induction on the structure of Q. □

Example 5.2. Let $Q_1 = a^+.\mathrm{Nil} + a^+.b^+.\mathrm{Nil}$ and $Q_2 = a^+.\mathrm{Nil}$; then

$$\mathrm{Beh}(Q_1) = \{\mathrm{Nil}, a.\mathrm{Nil}, a.b.\mathrm{Nil}, a.\mathrm{Nil} + a.b.\mathrm{Nil}\},$$

$$\mathrm{Beh}(Q_2) = \{\mathrm{Nil}, a.\mathrm{Nil}\}.$$

Note that an implicit specification $(Q, U) \in \mathcal{I}[(\mathrm{NEW}_{\mathcal{T}F}, \mathcal{T}F]$ is consistent (i.e. $Q \gg P \sim U$ for some process P) if and only if $U \sim R$ for some $R \in \mathrm{Beh}(Q)$.

We can now proceed to the statement of the main theorem, $\mathcal{H} \to \mathcal{I}[\mathrm{NEW}_{\mathcal{T}F}, \mathcal{T}F]$. The proof is given in full in Appendix A.

Theorem 5.3. *Let $F \in \mathcal{H}$. Then there exist $Q_F \in \mathcal{T}F$ and $P_F \in \mathcal{T}F$ such that the following equivalence holds for all processes P:*

$$P \models F \Leftrightarrow Q_F \gg P \sim P_F.$$

Proof. The full proof is given in Appendix A. In Table 2 we show the inductive definitions of Q_F and P_F. For $P \in \mathcal{T}F$, one can obtain a process $P^\circ \in \mathcal{T}F$ bisimilar to

Table 2
$\mathcal{H} \to \mathcal{T}[\mathrm{NEW}_{\mathcal{T}F}, \mathcal{T}F]$

F	Q_F	P_F
tt	Nil	Nil
ff	Nil	$a.\mathrm{Nil}$
$[a]G$	$a^+.Q_G + a^\circ.P_G$	$a.P_G$
$\langle a \rangle G$	$\sum_{i=2}^{n} a^+.R_i^G + a^+.Q_G$	$\sum_{i=2}^{n} a.R_i^G + a.P_G$
$H \wedge G$	$a^\circ.Q_H + b^\circ.Q_G$	$a.P_H + b.P_G$
$H \vee G$	$\sum_{i=2}^{m} x^\circ.V_i + \sum_{i=2}^{n} x^\circ.U_i + x^\circ.Q_1 + x^\circ.Q_2$ where $V_i = y.P_H + z.R_i^G + z.P_G$, $U_i = y.P_H + y.R_i^H + z.P_G$, $Q_1 = z^\circ.P_G + y^\circ.P_H + y^\circ.Q_H$, $Q_2 = z^\circ.P_G + y^\circ.P_H + z^\circ.Q_G$, $O = y.P_H + z.P_G$	$\sum_{i=2}^{m} x.V_i + \sum_{i=2}^{n} x.U_i + x.O$

$P[A^\circ/A]$ by simply substituting a° for any occurrence of an action a. In the cases $\langle a\rangle G$ and $H \vee G$, H and G are assumed to be consistent. Otherwise, the formula may be reduced to (is equivalent to) one of the smaller formulas H, G or ff for which a construction is already given. Also, in the definitions for $\langle a\rangle G$ and $H \vee G$, let

$$\mathrm{Beh}(Q_H)=\{R_1^H, \ldots, R_n^H\}, \qquad \mathrm{Beh}(Q_G)=\{R_1^G, \ldots, R_m^G\},$$

and assume, without loss of generality, that $P_H \sim R_1^H$ and $P_G \sim R_1^G$. That P_H (P_G) is bisimilar to some member of $\mathrm{Beh}(Q_H)$ ($\mathrm{Beh}(Q_G)$) follows from the assumed consistency of H and G (and the induction hypothesis). □

Example 5.4. In this example we shall use the constructions of Table 2 to generate implicit finite specifications equivalent to each of the Hennessy–Milner logic formulas $\langle b\rangle$tt, $[b]$ff, $\langle a\rangle\langle b\rangle$tt, $[a]\langle b\rangle$tt, $[a][b]$tt and $\langle b\rangle \mathrm{tt} \wedge [b]\mathrm{ff}$. Now, it is easy to see that $\mathrm{Beh}(Q_{\langle b\rangle \mathrm{tt}})=\mathrm{Beh}(Q_{[b]\mathrm{ff}})=\{\mathrm{Nil}, b.\mathrm{Nil}\}$. Hence, for $\langle b\rangle$tt and $[b]$ff, we obtain the following implicit specifications:[7]

$$Q_{\langle b\rangle \mathrm{tt}}=b^+.\mathrm{Nil}, \qquad Q_{[b]\mathrm{ff}}=b^+.\mathrm{Nil},$$

$$P_{\langle b\rangle \mathrm{tt}}=b.\mathrm{Nil}, \qquad P_{[b]\mathrm{ff}}=\mathrm{Nil}.$$

Thus, for $\langle a\rangle\langle b\rangle$tt, we obtain

$$Q_{\langle a\rangle\langle b\rangle \mathrm{tt}}=a^+.\mathrm{Nil}+a^+.b^+.\mathrm{Nil},$$

$$P_{\langle a\rangle\langle b\rangle \mathrm{tt}}=a.\mathrm{Nil}+a.b.\mathrm{Nil};$$

hence, according to Theorem 5.3 and using the various laws for $\gg$, we obtain

$$\begin{aligned} P &\models \langle a\rangle\langle b\rangle \mathrm{tt} \\ &\Leftrightarrow (a^+.\mathrm{Nil}+a^+.b^+.\mathrm{Nil}) \gg P \sim a.\mathrm{Nil}+a.b.\mathrm{Nil} \\ &\Leftrightarrow a.\mathrm{Nil} \parallel P + a.b.\mathrm{Nil} \parallel P \sim a.\mathrm{Nil}+a.b.\mathrm{Nil}. \end{aligned}$$

For $[a]\langle b\rangle$tt and $[a][b]$ff, we obtain the following implicit specification using the constructions of Table 2:

$$Q_{[a]\langle b\rangle \mathrm{tt}}=a^+.b^+\mathrm{Nil}+a^\circ.b^\circ.\mathrm{Nil}, \qquad Q_{[a][b]\mathrm{ff}}=a^+.b^+.\mathrm{Nil}+a^\circ.\mathrm{Nil},$$

$$P_{[a]\langle b\rangle \mathrm{tt}}=a.b.\mathrm{Nil}, \qquad P_{[a][b]\mathrm{ff}}=a.\mathrm{Nil}.$$

Then application of the algebraic laws for $\gg$ yields:

$$\begin{aligned} P &\models [a]\langle b\rangle \mathrm{tt} \\ &\Leftrightarrow (a^+.b^+.\mathrm{Nil}+a^\circ.b^\circ.\mathrm{Nil}) \gg P \sim a.b.\mathrm{Nil} \\ &\Leftrightarrow a.b.\mathrm{Nil} \parallel P + a.b.\mathrm{Nil} \sim a.b.\mathrm{Nil}. \end{aligned}$$

[7] The construction of Table 2 gives $Q_{[b]\mathrm{ff}}=b^+.\mathrm{Nil}+b^\circ.a^\circ.\mathrm{Nil}$ and $P_{[b]\mathrm{ff}}=b.a.\mathrm{Nil}$. In the example we use a slightly simpler (but obviously correct) implicit specification for $[b]$ff.

$$P \models [a][b]\mathrm{ff}$$
$$\Leftrightarrow (a^+.b^+.\mathrm{Nil} + a^\circ.\mathrm{Nil}) \gg P \sim a.\mathrm{Nil}$$
$$\Leftrightarrow a.b.\mathrm{Nil} \parallel P + a.\mathrm{Nil} \sim a.\mathrm{Nil}.$$

Finally, we obtain, for (the inconsistent) formula $\langle b\rangle \mathrm{tt} \wedge [b]\mathrm{ff}$,

$$Q_{\langle b\rangle \mathrm{tt} \wedge [b]\mathrm{ff}} = a^\circ.b^+.\mathrm{Nil} + b^\circ.b^+.\mathrm{Nil},$$
$$P_{\langle b\rangle \mathrm{tt} \wedge [b]\mathrm{ff}} = a.\mathrm{Nil} + b.b.\mathrm{Nil},$$

and, hence,

$$P \models \langle b\rangle \mathrm{tt} \wedge [b]\mathrm{ff}$$
$$\Leftrightarrow a.(b.\mathrm{Nil} \parallel P) + b.(b.\mathrm{Nil} \parallel P) \sim a.\mathrm{Nil} + b.b.\mathrm{Nil}.$$

That is, there is no process P that satisfies the above equation (which we hope the reader can see after a little reflection).

We conclude this section by stating as a main theorem the obtained characterization of the expressive power of implicit finite specifications.

Theorem 5.5. *For* $\mathrm{PAR}_{\mathcal{F}F}$ *ranging over* $\mathrm{NEW}_{\mathcal{F}F}$, $\mathrm{CCS}_{\mathcal{F}F}$ *and* $\mathrm{LOTOS}_{\mathcal{F}F}$, *following equivalences with respect to expressive power hold:*[8]

$$\mathcal{I}[\mathrm{PAR}_{\mathcal{F}F}, \mathcal{T}F] \leftrightarrow \mathcal{I}[\mathrm{PAR}_{\mathcal{F}F}, \mathcal{F}] \leftrightarrow \mathcal{I}[\mathrm{PAR}_{\mathcal{F}F}, \mathcal{H}] \leftrightarrow \mathcal{H}.$$

6. Implicit regular specifications

In this section we will demonstrate that implicit specifications induced by regular modal specifications $\mathcal{R}_c$ and the recursive extension of Hennessy–Milner logic $\mathcal{H}_v$ with respect to families of regular contexts (i.e. $\mathrm{PAR}_{\mathcal{F}}$, with PAR ranging over NEW, CCS and LOTOS; see Table 1) coincide with respect to expressive power. Moreover, we shall demonstrate that the common expressive power is precisely that of $\mathcal{H}_v$ itself. We have visualized these results of relative expressiveness in Fig. 5. From [21] we know that $\mathcal{I}[\mathcal{C}, \mathcal{H}_v] \rightarrow \mathcal{H}_v$ holds for any family of unary contexts $\mathcal{C}$ semantically describable as finite-state action transducers (and, hence, for the families $\mathrm{NEW}_{\mathcal{F}}$, $\mathrm{CCS}_{\mathcal{F}}$ and $\mathrm{LOTOS}_{\mathcal{F}}$). Using a slight extension of Lemma 4.1, it may be concluded that $\mathcal{I}[\mathrm{NEW}_{\mathcal{F}}, \mathcal{R}_c]$ is no more expressive than $\mathcal{I}[\mathrm{CCS}_{\mathcal{F}}, \mathcal{R}_c]$ and $\mathcal{I}[\mathrm{LOTOS}_{\mathcal{F}}, \mathcal{R}_c]$. Thus, it suffices to consider the family $\mathrm{NEW}_{\mathcal{F}}$.

In [18] we proved that $\mathcal{H} \rightarrow \mathcal{I}[\mathcal{C}, \mathcal{F}]$, where $\mathcal{C}$ was a "sufficiently rich" family of contexts. Here we extend this result to the recursive/regular case. More precisely, we show that

$$\mathcal{H}_v \rightarrow \mathcal{I}[\mathrm{NEW}_{\mathcal{F}}, \mathcal{R}_c].$$

[8] $\leftrightarrow$ denotes the equivalence generated by $\rightarrow$, i.e. $\mathcal{S}_1 \leftrightarrow \mathcal{S}_2$ if and only if $\mathcal{S}_1 \rightarrow \mathcal{S}_2$ and $\mathcal{S}_2 \rightarrow \mathcal{S}_1$.

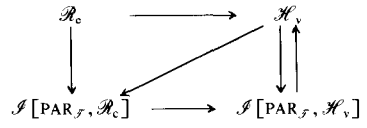

Fig. 5. Relative expressiveness II.

Theorem 6.1. *Let $F \in \mathscr{H}_v$. Then there exist $P_F \in \mathscr{T}$ and $S_F \in \mathscr{R}_c$ such that the following equivalence holds for all processes P:*

$$P \models F \Leftrightarrow (P_F \gg P) \lhd S_F.$$

Proof. The full proof is given in Appendix B. In Table 3 we show the inductive definitions of P_F and S_F. □

Example 6.2. In this example we shall use the constructions of Table 3 to generate implicit regular specifications from (nonrecursive) Hennessy–Milner logic formulas. First, for $\langle a \rangle \mathrm{tt}$ and $[a]\mathrm{ff}$, we obtain the following implicit specifications:

$$P_{\langle a \rangle \mathrm{tt}} = a^+.\mathrm{Nil}, \qquad P_{[a]\mathrm{ff}} = a^+.\mathrm{Nil},$$

$$S_{\langle a \rangle \mathrm{tt}} = a_{\Box}.\mathrm{Nil} + a_{\Diamond}.\mathscr{U}, \qquad S_{[a]\mathrm{ff}} = a_{\Diamond}.a_{\Box}.\mathrm{Nil}.$$

Hence, for the inconsistent (nonrecursive) formula $\langle a \rangle \mathrm{tt} \wedge [a]\mathrm{ff}$, we obtain:[9]

$$P_{\langle a \rangle \mathrm{tt} \wedge [a]\mathrm{ff}} = a^{\circ}.a^+.\mathrm{Nil} + b^{\circ}.a^+.\mathrm{Nil},$$

$$S_{\langle a \rangle \mathrm{tt} \wedge [a]\mathrm{ff}} = a_{\Diamond}.(a_{\Box}.\mathrm{Nil} + a_{\Diamond}.\mathscr{U}) + b_{\Diamond}.a_{\Diamond}.a_{\Box}.\mathrm{Nil},$$

and, hence, according to Theorem 6.1 and using the various laws for $\gg$, we obtain

$$P \models \langle a \rangle \mathrm{tt} \wedge [a]\mathrm{ff}$$

$$\Leftrightarrow a.(a.\mathrm{Nil} \parallel P) + b.(a.\mathrm{Nil} \parallel P) \lhd a_{\Diamond}.(a_{\Box}.\mathrm{Nil} + a_{\Diamond}.\mathscr{U}) + b_{\Diamond}.a_{\Diamond}.a_{\Box}.\mathrm{Nil}.$$

Table 3
$\mathscr{H}_v \to \mathscr{I}[\mathrm{NEW}_{\mathscr{F}}, \mathscr{R}_c]$

F	P_F	S_F
tt	Nil	Nil
ff	Nil	$a_{\Box}.\mathrm{Nil}$
$H \wedge G$	$a^{\circ}.P_H + b^{\circ}.P_G$	$a_{\Diamond}.S_H + b_{\Diamond}.S_G$
$H \vee G$	$a^{\circ}.a^{\circ}.P_H + a^{\circ}.b^{\circ}.P_G$	$a_{\Box}.(a_{\Diamond}.S_H + b_{\Diamond}.S_G) + a_{\Diamond}.\mathscr{U}$
$\langle a \rangle H$	$a^+.P_H$	$a_{\Box}.S_H + a_{\Diamond}.\mathscr{U}$
$[a]H$	$a^+.P_H$	$a_{\Diamond}.S_H$
x	x	x
$\nu x.F$	$\mathrm{rec}x.P_F$	$\mathrm{rec}x.S_F$

[9] As $\langle a \rangle \mathrm{tt} \wedge [a]\mathrm{ff}$ is inconsistent, it is, of course, not directly expressible in $\mathscr{R}_c$.

Now consider the formula $\langle a\rangle \mathrm{tt} \vee \langle b\rangle \mathrm{tt}$. From the results of [4] it can be concluded that this formula is not directly expressible in $\mathscr{R}_c$. However, using implicit specifications yields the following:

$$P_{\langle a\rangle \mathrm{tt} \vee \langle b\rangle \mathrm{tt}} = a^{\circ}.a^{\circ}.a^{+}.\mathrm{Nil} + a^{\circ}.b^{\circ}.b^{+}.\mathrm{Nil},$$

$$S_{\langle a\rangle \mathrm{tt} \vee \langle b\rangle \mathrm{tt}} = a_{\sqcup}.(a_{\diamond}.(a_{\_}.\mathrm{Nil} + a_{\diamond}.\mathscr{U}) + b_{\diamond}.(b_{\sqcup}.\mathrm{Nil} + b_{\diamond}.\mathscr{U})) + a_{\diamond}.\mathscr{U}.$$

Hence, from Theorem 6.1 we obtain

$$\begin{aligned} P \models \langle a\rangle \mathrm{tt} \vee \langle b\rangle \mathrm{tt} \\ \Leftrightarrow\ & a.a.(a.\mathrm{Nil} \,\|\, P) + a.b.(b.\mathrm{Nil} \,\|\, P) \lhd a_{\sqcup}.(a_{\diamond}.(a_{\_}.\mathrm{Nil} + a_{\diamond}.\mathscr{U}) \\ & + b_{\diamond}.(b_{\sqcup}.\mathrm{Nil} + b_{\diamond}.\mathscr{U})) + a_{\diamond}.\mathscr{U}. \end{aligned}$$

For formulas of the form $\langle a\rangle \mathrm{tt} \wedge [a]F$, the construction of Theorem 6.1 can be "optimized" as follows:

$$P_{\langle a\rangle \mathrm{tt} \wedge [a]F} = a^{+}.P_F, \qquad S_{\langle a\rangle \mathrm{tt} \wedge [a]F} = a_{\sqcup}.S_F.$$

With this optimization we can represent the (recursive) safety properties of Example 3.8 as implicit specifications as follows.

Example 6.3. First consider the property an infinite a-computation $\nu x.\langle a\rangle x$. Here

$$P_{\nu x.\langle a\rangle x} = \mathrm{rec}x.a^{+}.x, \qquad S_{\nu x.\langle a\rangle x} = \mathrm{rec}x.(a_{\_}.x + a_{\diamond}.\mathscr{U}).$$

Thus, according to Theorem 6.1 and using the algebraic laws for $\gg$,

$$\begin{aligned} P \models \nu x.\langle a\rangle x \\ \Leftrightarrow\ & \mathrm{rec}x\ .a.x \,\|\, P \lhd \mathrm{rec}x.(a_{\_}.x + a_{\diamond}.\mathscr{U}). \end{aligned}$$

The absence of deadlock under a-actions can be formulated as the invariance property $\nu x.\langle a\rangle \mathrm{tt} \wedge [a]x$. Using Theorem 6.1, we can formulate this property as an implicit specification as follows:

$$P_{\nu x.\langle a\rangle \mathrm{tt} \wedge [a]x} = \mathrm{rec}x.a^{+}.x, \qquad S_{\nu x.\langle a\rangle \mathrm{tt} \wedge [a]x} = \mathrm{rec}x.a_{\_}.x.$$

Hence,

$$\begin{aligned} P \models \nu x.\langle a\rangle \mathrm{tt} \wedge [a]x \\ \Leftrightarrow\ & \mathrm{rec}x.a.x \,\|\, P \lhd \mathrm{rec}x.a_{\sqcup}.x \\ \Leftrightarrow\ & \mathrm{rec}x.a.x \,\|\, P \sim \mathrm{rec}x.a.x \end{aligned}$$

since $\mathrm{rec}x.a_{\sqcup}.x \in \mathscr{T}$. Finally, the property that all finite a-computations have even length ($\nu x.[a](\langle a\rangle \mathrm{tt} \wedge [a]x)$) can be represented as an implicit specification as follows:

$$P_{\nu x.[a](\langle a\rangle \mathrm{tt} \wedge [a]x)} = \mathrm{rec}x.a^{+}.a^{+}.x, \qquad S_{\nu x.[a](\langle a\rangle \mathrm{tt} \wedge [a]x)} = \mathrm{rec}x.a_{\diamond}.a_{\sqcup}.x.$$

Thus,

$$P \models \nu x.[a](\langle a\rangle \mathrm{tt} \wedge [a]x)$$
$$\Leftrightarrow \mathrm{rec}x.a.a.x \,\|\, P \lhd \mathrm{rec}x.a_{\Diamond}.a_{\Box}.x.$$

We may now, as a main theorem, state the obtained characterization of the expressive power of implicit regular specifications.

Theorem 6.4. *For* $\mathrm{PAR}_{\mathcal{F}}$ *ranging over* $\mathrm{NEW}_{\mathcal{F}}$, $\mathrm{CCS}_{\mathcal{F}}$ *and* $\mathrm{LOTOS}_{\mathcal{F}}$, *the following equivalences with respect to expressive power hold*:

$$\mathcal{I}[\mathrm{PAR}_{\mathcal{F}}, \mathcal{R}_{\mathrm{c}}] \leftrightarrow \mathcal{I}[\mathrm{PAR}_{\mathcal{F}}, \mathcal{H}_{\mathrm{v}}] \leftrightarrow \mathcal{H}_{\mathrm{v}}.$$

Finally, we conclude this section by observing an immediate improvement of the lower-bound complexity results of [15].

Corollary 6.5. *The consistency problem for* $\mathcal{I}[\mathrm{CCS}_{\mathcal{F}}, \mathcal{R}_{\mathrm{c}}]$ *is* DEXPTIME-*complete.*

Proof. From the results of [10] it follows that the satisfiability (and validity) problem for $\mathcal{H}_{\mathrm{v}}$ is DEXPTIME-complete. The translation $\mathcal{H}_{\mathrm{v}} \rightarrow \mathcal{I}[\mathrm{NEW}_{\mathcal{F}}, \mathcal{R}_{\mathrm{c}}]$ in Table 3, obviously, yields an implicit specification with polynomial size in terms of the initial formula; hence, we may conclude that the consistency problem for $\mathcal{I}[\mathrm{NEW}_{\mathcal{F}}, \mathcal{R}_{\mathrm{c}}]$ is also DEXPTIME-complete. Theorem 4.2 provides the basis for extending the complexity result to $\mathcal{I}[\mathrm{CCS}_{\mathcal{F}}, \mathcal{R}_{\mathrm{c}}]$. □

7. Conclusion

The results of this paper provide an insight into the contest *behavioural versus logical specification formalisms*. In general, behavioural specifications are strictly less expressive than logical ones. However, resorting to *implicit* behavioural specifications gives one the full expressive power of a logical specification formalism with a maximal fixed-point construct. Thus, implicit behavioural specifications are useful for expressing *safety* properties and may be combined freely under logical connectives such as conjunction and disjunction.

The paper leaves as an open problem the characterization of the expressive power of implicit specifications of the form

$$(A|P[\Phi])\backslash L \sim B,$$

where A and B are *regular processes*. That is, what is the expressive power of the implicit specification formalism $\mathcal{I}[\mathrm{CCS}_{\mathcal{F}}, \mathcal{T}]$? Based on the monotonicity of $\mathcal{I}[,]$ and Main Theorem 6.1, we know that $\mathcal{I}[\mathrm{CCS}_{\mathcal{F}}, \mathcal{T}]$ can be no more expressive than $\mathcal{H}_{\mathrm{v}}$. The question is whether $\mathcal{I}[\mathrm{CCS}_{\mathcal{F}}, \mathcal{T}]$ equals $\mathcal{H}_{\mathrm{v}}$ in expressive power. Related to

this question is the question of complexity for the consistency problem of $\mathscr{I}[\text{CCS}_{\mathscr{F}}, \mathscr{T}]$. We know from [15] that this decision problem is PSPACE-hard, but whether it is in PSPACE or is an inherent DEXPTIME-problem is still an open problem.

Future work should include investigating the expressive power of implicit specifications based on other equivalences (in particular, that of observational equivalence), and implicit specifications of several components, i.e. simultaneous specifications of n processes $P_1, \ldots, P_n$ of the form

$$C(P_1, \ldots, P_n) \textbf{ sat } S.$$

The problem left open by this paper is the presence of minimal fixedpoints in the logical formalism. It would be interesting to characterize the type of contexts needed in order to recover expressibility of liveness properties. Also, the paper leaves as an open problem the treatment of weak equivalences, which, from practical design situations, are clearly very important.

Finally, it seems that the notion of "implicit specification" is completely general and applicable to other specification formalisms.

Appendix A. Characterization of implicit finite specifications

In this appendix we give the full proof of the Main Theorem 5.3.

Theorem A.1. *Let $F \in \mathscr{H}$. Then there exist $Q_F \in \mathscr{T}F$ and $P_F \in \mathscr{T}F$ such that the following equivalence holds:*

$$\forall P.\ P \models F \Leftrightarrow (Q_F \gg P) \sim P_F.$$

Proof. Q_F and P_F are defined in Table 2 by induction on F. Here we argue inductively that these definitions will satisfy the theorem.

$F = \text{tt}$ *and* $F = \text{ff}$: With the definitions of $Q_{\text{tt}}, P_{\text{tt}}$ and $Q_{\text{ff}}, P_{\text{ff}}$ in Table 2, it is obvious that $(Q_F \gg P) \sim P_F$ holds if and only if $P \models F$.

$F = [a]G$: Recall from Table 2 that

$$Q_{[a]G} = a^+.Q_G + a^\circ.P_G^\circ, \qquad P_{[a]G} = a.P_G.$$

Using the various laws for $\gg$, we can prove the desired induction step as follows:

$$\begin{aligned}
&(Q_{[a]G} \gg P) \sim P_{[a]G} \\
&\quad \Leftrightarrow (a^+.Q_G + a^\circ.P_G^\circ) \gg P \sim P_{[a]G} \\
&\quad \Leftrightarrow (a^+.Q_G \gg P) + (a^\circ.P_G^\circ \gg P) \sim P_{[a]G} \\
&\quad \Leftrightarrow (a^+.Q_G \gg P) + a.P_G \sim a.P_G
\end{aligned}$$

$$\Leftrightarrow \text{ whenever } (a^+.Q_G \gg P) \xrightarrow{a} R,\ R \sim P_G \tag{14}$$

$$\Leftrightarrow \text{ whenever } P \xrightarrow{a} P',\ Q_G \gg P' \sim P_G \tag{15}$$

$$\Leftrightarrow \text{ whenever } P \xrightarrow{a} P',\ P' \models G \tag{16}$$

$$\Leftrightarrow P \models [a]G.$$

In (14) only one-half of the proof obligations for bisimulation is needed as the transition of the right-hand process clearly can be matched by the left-hand process. In (15) we are simply using the inference rules for $\gg$, and (16) is a simple application of the induction hypothesis.

$F = \langle a \rangle G$: If G is inconsistent (i.e. $P \models G$ for no process P), also $\langle a \rangle G$ is inconsistent and we can simply let $Q_F = Q_{\mathrm{ff}}$ and $P_F = P_{\mathrm{ff}}$ in order to satisfy the statement of the theorem. Otherwise, $P \models G$ or (using the induction hypothesis) $Q_G \gg P \sim P_G$ for some process P. Now let $\mathrm{Beh}(Q_G) = \{R_1^G, \ldots, R_n^G\}$; then $P_G \sim R_i^G$ for some i. Without loss of generality, we shall assume $P_G \sim R_1^G$. Now recall the following from Table 2:

$$Q_{\langle a \rangle G} = \sum_{i=2}^{n} a^+.R_i^G + a^+.Q_G,$$

$$P_{\langle a \rangle G} = \sum_{i=2}^{n} a.R_i^G + a.P_G.$$

We now establish the induction step.

$\Rightarrow$: Assume $(Q_{\langle a \rangle G} \gg P) \sim P_{\langle a \rangle G}$. In order for the transition $P_{\langle a \rangle G} \xrightarrow{a} P_G$ to be matched, P must have a transition $P \xrightarrow{a} P'$ such that either $Q_G \gg P' \sim P_G$ or, for some $i = 2, \ldots, n$, $R_i^G \gg P' \sim P_G$. However, using the strengthened property of $\mathrm{Beh}(Q_G)$ and the algebraic law in (11),

$$R_i^G \gg P' \sim R_i^G \not\sim R_1^G \sim P_G$$

for $i = 2, \ldots, n$. Thus, $P \xrightarrow{a} P'$ with $Q_G \gg P' \sim P_G$ or (using the induction hypothesis) $P' \models G$ and, hence, $P \models \langle a \rangle G$.

$\Leftarrow$: Now assume $P \models \langle a \rangle G$. Thus, for some P' with $P \xrightarrow{a} P'$, $P' \models G$ or, equivalently (using the induction hypothesis), $Q_G \gg P' \sim P_G$. With this knowledge, we may now argue for the desired equivalence $Q_{\langle a \rangle G} \gg P \sim P_{\langle a \rangle G}$. First consider the possible transitions of $Q_{\langle a \rangle G} \gg P$:

(i) $Q_{\langle a \rangle G} \gg P \xrightarrow{a} R_i^G \gg P''$ $(i = 2, \ldots, n)$,

(ii) $Q_{\langle a \rangle G} \gg P \xrightarrow{a} Q_G \gg P''$,

where $P \xrightarrow{a} P''$. According to the algebraic law (11), $R_i^G \gg P'' \sim R_i^G$ and, hence, $P_{\langle a \rangle G} \xrightarrow{a} R_i^G$ clearly provides a match for case (i). As $\{P_G, R_2^G, \ldots, R_n^G\} = \mathrm{Beh}(Q_G)$, it follows in case (ii) that $P_{\langle a \rangle G}$ has a transition matching $Q_G \gg P''$ for any P''. Now consider the possible transitions of $P_{\langle a \rangle G}$:

(i) $P_{\langle a \rangle G} \xrightarrow{a} R_i^G \quad (i = 2, \ldots, n)$,

(ii) $P_{\langle a \rangle G} \xrightarrow{a} P_G$.

We already know that $P \xrightarrow{a} P'$ with $Q_G \gg P' \sim P_G$; hence, $Q_{\langle a \rangle G} \gg P \xrightarrow{a} R_i^G \gg P' \sim R_i^G$ clearly matches (i) and $Q_{\langle a \rangle G} \gg P \xrightarrow{a} Q_G \gg P'$ clearly matches (ii).

$F = H \wedge G$: From the inference rules of $\gg$ it follows immediately that $Q_{H \wedge G} \gg P \sim P_{H \wedge G}$ if and only if $Q_H \gg P \sim P_H$ and $Q_G \gg P \sim P_H$. Using the induction hypothesis, this is equivalent to $P \models H$ and $P \models G$ or $P \models H \wedge G$.

$F = H \vee G$: We may assume that both H and G are consistent. Otherwise, $H \vee G$ can be reduced to one of the subformulas H and G, formulas for which the theorem has already been established according to the induction hypothesis. Thus, let $\mathrm{Beh}(Q_H) = \{R_1^H, \ldots, R_n^H\}$ with $P_H \sim R_1^H$ and, similarly, let $\mathrm{Beh}(Q_G) = \{R_1^G, \ldots, R_m^G\}$ with $P_G \sim R_1^G$. Now recall the constructions of $Q_{H \vee G}$ and $P_{H \vee G}$ from Table 2. First we show that

$$Q_{H \vee G} \gg P \sim P_{H \vee G} \tag{17}$$
$$\Leftrightarrow \; Q_1 \gg P \sim O \text{ or } Q_2 \gg P \sim O.$$

(17)$\Rightarrow$: To match $P_{H \vee G} \xrightarrow{x} O$ neither $Q_{H \vee G} \gg P \xrightarrow{x} V_i^\circ \gg P$ nor $Q_{H \vee G} \gg P \xrightarrow{x} U_i^\circ \gg P$ will do as $V_i^\circ \gg P \sim V_i$ ($U_i^\circ \gg P \sim U_i$) are not equivalent to O. To see why $V_i \not\sim O$, simply observe that $V_i \xrightarrow{z} R_i^G$, for which the only possible match is $O \xrightarrow{z} P_G$. However, $P_G \sim R_1^G \not\sim R_i^G$ for $i = 2, \ldots, n$ due to the strengthened property of $\mathrm{Beh}(Q_G)$. This leaves $Q_{H \vee G} \gg P \xrightarrow{x} Q_1 \gg P$ and $Q_{H \vee G} \gg P \xrightarrow{x} Q_2 \gg P$ as possible matches.

(17)$\Leftarrow$: We list the possible transitions and their matches under the assumption that $Q_1 \gg P \sim O$ or $Q_2 \gg P \sim O$:

Transition	Match
$P_{H \vee G} \xrightarrow{x} V_i$	$Q_{H \vee G} \gg P \xrightarrow{x} V_i^\circ \gg P$
$P_{H \vee G} \xrightarrow{x} U_i$	$Q_{H \vee G} \gg P \xrightarrow{x} U_i^\circ \gg P$
$P_{H \vee G} \xrightarrow{x} O$	$Q_{H \vee G} \gg P \xrightarrow{x} Q_1 \gg P$ or $Q_{H \vee G} \gg P \xrightarrow{x} Q_2 \gg P$

$$Q_{H\vee G} \gg P \xrightarrow{x} V_i^\circ \gg P \qquad P_{H\vee G} \xrightarrow{x} V_i$$

$$Q_{H\vee G} \gg P \xrightarrow{x} U_i^\circ \gg P \qquad P_{H\vee G} \xrightarrow{x} U_i \tag{18}$$

$$Q_{H\vee G} \gg P \xrightarrow{x} Q_1 \gg P \qquad \begin{cases} P_{H\vee G} \xrightarrow{x} O, & \text{if } Q_1 \gg P \sim O \\ P_{H\vee G} \xrightarrow{x} U_j, & \text{if } R_j^H \sim Q_H \gg P \end{cases} \tag{19}$$

$$Q_{H\vee G} \gg P \xrightarrow{x} Q_2 \gg P \qquad \begin{cases} P_{H\vee G} \xrightarrow{x} O, & \text{if } Q_2 \gg P \sim O \\ P_{H\vee G} \xrightarrow{x} V_j, & \text{if } R_i^G \sim Q_G \gg P \end{cases} \tag{20}$$

Due to the definition of Beh, we know in (18) (and, similarly, in (19)) that $R_j^H \sim Q_H \gg P$ is satisfied for some j. To see that U_j then indeed does match $Q_1 \gg P$ note that (using the various laws for $\gg$)

$$\begin{aligned} &Q_1 \gg P \\ &= (z^\circ . P_G^\circ + y^\circ . P_H^\circ + y^\circ . Q_H) \gg P \\ &\sim z . P_G + y . P_H + y . (Q_H \gg P) \\ &\sim z . P_G + y . P_H + y . R_j^H \\ &= U_j . \end{aligned}$$

To complete the proof we show that

$$Q_1 \gg P \sim O \;\Leftrightarrow\; Q_H \gg P \sim P_H \tag{21}$$

(a similar statement can be proved for G).

(21)$\Rightarrow$: This direction is obvious as the transition $Q_1 \gg P \xrightarrow{y} Q_H \gg P$ can only possibly be matched by $O \xrightarrow{y} P_H$.

(21)$\Leftarrow$: We list the transitions and their (possible) matches under the assumption that $Q_H \gg P \sim P_H$:

Transition	Match
$O \xrightarrow{y} P_H$	$Q_1 \gg P \xrightarrow{y} Q_H \gg P$
$O \xrightarrow{z} P_G$	$Q_1 \gg P \xrightarrow{z} P_G^\circ \gg P$
$Q_1 \gg P \xrightarrow{z} P_G^\circ \gg P$	$O \xrightarrow{z} P_G$
$Q_1 \gg P \xrightarrow{y} P_H^\circ \gg P$	$O \xrightarrow{y} P_H$
$Q_1 \gg P \xrightarrow{y} Q_H \gg P$	$O \xrightarrow{y} P_H$

From (17), (21) and the induction hypothesis the induction step follows. □

Appendix B. Characterization of implicit regular specifications

In this appendix we sketch a full proof of the Main Theorem 6.1. Throughout the appendix we assume, for $F \in \mathscr{L}$, that P_F and S_F are defined according to Table 3. In particular, note that F, P_F and S_F have identical sets of free variables. We claim the following generalization of Theorem 6.1 (generalized to formulas with free variables).

Theorem B.1. *Let $F \in \mathscr{L}$ and let $x = \{x_1, \ldots, x_n\}$ contain all free variables of F. Then, for any process P,*[10] *the following holds:*

$$(P_F[Q/x] \gg P) \lhd S_F[S/x] \Leftrightarrow P \in [\![F]\!]\{U/x\},$$

where $Q = \{Q_1, \ldots, Q_n\}$ are closed regular process expressions (i.e. from $\mathscr{T}$), $S = \{S_1, \ldots, S_n\}$ are closed regular modal specifications (i.e. from $\mathscr{R}_c$) and $U = \{U_1, \ldots, U_n\}$, with

$$U_i = \{R \in \Pi \mid (Q_i \gg R) \lhd S_i\}.$$

For closed formula, Theorem B.1 clearly yields Theorem 6.1 as a corollary.

Corollary B.2. *Let $F \in \mathscr{H}_v$. Then, for any process P, the following holds:*

$$P \models F \Leftrightarrow (P_F \gg P) \lhd S_F.$$

Proof of Theorem B.1. The proof is by induction on the structure of F. Here we only consider the cases $F = x$ and $F = \nu x.G$ and refer the reader to [18] for the remaining cases.

$F = x$: Using the definitions of P_x, S_x and U, the theorem is obvious in this case.

$F = \nu x.G$: Let ρ be the environment $\{U/x\}$, where $U_i = \{R \mid Q_i \gg R \lhd S_i\}$.

$\Rightarrow$: Define the set of processes V as follows:

$$V = \{P \mid P_{\nu x.G}[Q/x] \gg P \lhd S_{\nu x.G}[S/x]\}.$$

We will show that V satisfies

$$V \subseteq [\![G]\!]\rho\{V/x\}$$

and, hence, that $V \subseteq [\![\nu x.G]\!]\rho$. In other words, if $P \in V$ then $P \in [\![\nu x.G]\!]\rho$:

$$\begin{aligned} & P \in V \\ & \Leftrightarrow P_{\nu x.G}[Q/x] \gg P \lhd S_{\nu x.G}[S/x] \\ & \Leftrightarrow (\mathrm{rec}x.P_G)[Q/x] \gg P \lhd (\mathrm{rec}x.S_G)[S/x] \\ & \Leftrightarrow (\mathrm{rec}x.P_G[Q/x]) \gg P \lhd (\mathrm{rec}x.S_G[S/x]) \end{aligned}$$

[10] We shall, in this proof sketch, assume that P is *image-finite* in the sense that $\{Q' \mid Q \xrightarrow{a} Q'\}$ is a finite set for any derivative Q of P and any action a.

$$\Leftrightarrow P_G[Q/x][\mathrm{rec}x.P_G[Q/x]/x] \gg P \lhd S_G[S/x][\mathrm{rec}x.S_G[S/x]/x]$$

$$\Leftrightarrow P_G[Q/x][P_{\nu x.G}[Q/x]/x] \gg P \lhd S_G[S/x][S_{\nu x.G}[S/x]/x] \quad (22)$$

$$\Leftrightarrow P \in [\![G]\!]\rho\{V/x\}, \quad (23)$$

where in (22) we apply the induction hypothesis on G.

$\Leftarrow$: Assume that $P \in [\![\nu x.G]\!]\rho$. We must show that

$$P_{\nu x.G}[Q/x] \gg P \lhd S_{\nu x.G}[S/x].$$

Under the assumption of image-finiteness of P, $P \in [\![\nu x.G]\!]\rho$ is equivalent to

$$\forall n \in \omega\ P \in [\![G^n]\!]\{U/x, \Pi/x\},$$

where $G^0 = x$ and $G^{n+1} = G^n[G/x]$. We claim that this is equivalent to

$$\forall n \in \omega\ (P_{G^n}[Q/x, P_{\mathrm{tt}}/x] \gg P) \lhd S_{G^n}[S/x, S_{\mathrm{tt}}/x]. \quad (24)$$

This may be established by an easy (inner) induction on n. Under the assumption that P is image-finite, knowing that P_G, S_G are regular (and, hence, image-finite) for any G and that $\gg$ preserves image-finiteness, and assuming that x is *guarded* with respect to $\langle a \rangle$ or $[a]$ in $\nu x.G$ (and, hence, guarded in $P_{\nu x.G}$ and $S_{\nu x.G}$),[11] (24) becomes equivalent to

$$\forall n \in \omega\ (\mathrm{rec}x.P_G[Q/x] \gg P) \lhd^n \mathrm{rec}x.S_G[S/x],$$

where $\lhd^n$ is *refinement up to level n* (see [19]). Under the assumption of image-finiteness, it can be shown that $\lhd = \bigcap_{n \in \omega} \lhd^n$. Hence, it follows, as desired, that

$$(P_{\nu x.G}[Q/x] \gg P) \lhd S_{\nu x.G}[S/x]. \qquad \square$$

Acknowledgment

The author is most grateful to Ed Brinksma, Bengt Jonsson and Joachim Parrow for many valuable discussions on the problems of this paper. In particular, the algebraic law (13) is due to Ed Brinksma. Thanks are also due to the anonymous referees for their valuable comments. The work reported in this paper was carried out during a very pleasant sabbatical at the Swedish Institute of Computer Science.

References

[1] A. Arnold and P. Crubille, A linear algorithm to solve fixed-point equations on transition systems, *Inform. Process. Lett.* **29** (1988) 57–66.

[11] We can always transform an $\mathcal{L}$-formula into an $\langle a \rangle$- or $[a]$-guarded formula using the obvious equivalences between formulas: $\nu x.(G \vee x) \equiv \mathrm{tt}$ and $\nu x.(G \wedge x) \equiv \nu x.G$.

[2] M. Ben-Ari, A. Pnueli and Z. Manna, The temporal logic of branching time, *Acta. Inform.* **20** (1983) 207–226.

[3] T. Bolognesi and E. Brinksma, Introduction to the ISO specification language LOTOS, *Comput. Networks ISDN Systems* **14** (1987) 25–59.

[4] G. Boudol and K.G. Larsen, Graphical versus logical specifications, in: *Proc. Coll. on Trees in Algebra and Programming '90*, Lecture Notes in Computer Science, Vol. 431 (Springer, Berlin, 1990).

[5] J. Bradfield and C. Stiring, Verifying temporal properties of processes, Lecture Notes in Computer Science, Vol. 458 (Springer, Berlin, 1990).

[6] R. Cleaveland, Tableau-based model checking in the propositional mu-calculus, *Acta Inform.* **27** (1990) 725–747.

[7] R. Cleaveland and B. Steffen, Computing behavioural relations, logically, in: *Proc. 18th Internat. Coll. on Automata, Languages and Programming*, Lecture Notes in Computer Science, Vol. 510 (Springer, Berlin, 1991).

[8] R. de Simone, Higher-level synchronising devices in MEIJE-CCS, *Theoret. Comput. Sci.* **37** (1985) 245–267.

[9] E.A. Emerson and C.L. Lei, Efficient model checking in fragments of the propositional mu-calculus, in: *Proc. Logic in Computer Science* (1986) 267–278.

[10] M.J. Fischer and R.E. Ladner, Propositional dynamic logic of regular programs, *J. Comput. System Sci.* **18** (1979) 194–211.

[11] M. Hennessy and R. Milner, Algebraic laws for nondeterminism and concurrency, *J. ACM* **32** (1985) 137–161.

[12] C.A.R. Hoare, *Communicating Sequential Processes* (Prentice-Hall, Englewood Cliffs, NJ, 1985).

[13] C.A.R. Hoare and E.R. Olderog, Specification oriented semantics for communicating processes, in: *Proc. 10th Internat. Coll. on Automata, Languages and Programming*, Lecture Notes in Computer Science, Vol. 154 (Springer, Berlin, 1983).

[14] H. Hüttel and K.G. Larsen, The use of static constructs in a modal process logic, Lecture Notes in Computer Science, Vol. 363 (Springer, Berlin, 1989).

[15] B. Jonsson and K.G. Larsen, On the complexity of equation solving in process algebra, in: *Proc. TAPSOFT '91*, Lecture Notes in Computer Science, Vol. 493 (Springer, Berlin, 1991).

[16] D. Kozen, Results on the propositional mu-calculus, in: *Proc. Internat. Coll. on Algorithms, Languages and Programming 1982*, Lecture Notes in Computer Science, Vol. 140 (Springer, Berlin, 1982).

[17] K.G. Larsen, Proof systems for Hennessy–Milner logic with recursion, in: *Proc. 13th Coll. on Trees in Algebra and Programming 1988*, Lecture Notes in Computer Science, Vol. 299 (Springer, Berlin, 1988).

[18] K.G. Larsen, Ideal specification formalism = expressivity + compositionality + decidability + testability + ···, Invited paper at CONCUR '90, Lecture Notes in Computer Science, Vol. 458 (Springer, Berlin, 1990).

[19] K.G. Larsen, Modal specifications, in: *Proc. Workshop on Automatic Verification Methods for Finite State Systems*, Grenoble 1989, Lecture Notes in Computer Science, Vol. 407 (Springer, Berlin, 1990).

[20] K.G. Larsen and B. Thomsen, A modal process logic, in: *Proc. Logic in Computer Science* (1988) 203–211.

[21] K.G. Larsen and L. Xinxin, Compositionality through an operational semantics of contexts, in: *Proc. Internat. Coll. on Algorithms, Languages and Programming 1990*, Lecture Notes in Computer Science, Vol. 443 (Springer, Berlin, 1990).

[22] K.G. Larsen and L. Xinxin, Equation solving using modal transition systems, in: *Proc. Logic in Computer Science* (1990) 108–117.

[23] P. Lewis and H. Qin, Factorization of finite state machines under observational equivalence, Lecture Notes in Computer Science, Vol. 458 (Springer, Berlin, 1990).

[24] P. Merlin and G. von Bochmann, On the construction of submodule specifications and communication protocols, *ACM Trans. Programming Languages Systems* **5**(1) (1983) 1–25.

[25] R. Milner, *Calculus of Communicating Systems*, Lecture Notes in Computer Science, Vol. 92 (Springer, Berlin, 1980).

[26] R. Milner, Calculi for synchrony and asynchrony, *Theoret. Comput. Sci.* **25** (1983) 267–310.

[27] R. Milner, A complete inference system for a class of regular behaviours, *J. Comput. Sci. System* **28** (1984).

[28] R. Milner, *Communication and Concurrency* (Prentice-Hall, Englewood Cliffs, NJ, 1989).

[29] D. Park, Concurrency and automata on infinite sequences, in: *Proc. 5th GI Conf.*, Lecture Notes in Computer Science, Vol. 104 (Springer, Berlin, 1981).

[30] J. Parrow, Submodule construction as equation solving in CCS, *Theoret. Comput. Sci.* **68** (1989) 175–202.

[31] G. Plotkin, A structural approach to operational semantics, FN 19, DAIMI, Aarhus University, 1981.

[32] A. Pnueli, Linear and branching structures in the semantics and logics of reactive systems, in: *Proc. 12th Internat. Coll. on Automata, Languages and Programming*, Lecture Notes in Computer Science, Vol. 194 (Springer, Berlin, 1985).

[33] M.W. Shields, A note on the simple interface equation, Tech. Report, University of Kent at Canterbury.

[34] C. Stirling and D. Walker, Local model checking in the modal mu-calculus, in: *Proc. Tapsoft '89*, Lecture Notes in Computer Science, Vol. 352 (Springer, Berlin, 1989).

[35] G. Winskel, Model checking the modal nu-calculus, in: *Proc. Internat. Coll. on Algorithms, Languages and Programming 1989*, Lecture Notes in Computer Science, Vol. 372 (Springer, Berlin, 1989).